

**Bansilal Ramnath Agarwal Charitable Trust’s**

Vishwakarma Institute of Technology, Pune-37

*(Anautonomous Institute of Savitribai Phule Pune University)*

**Department of Computer Engineering**

|  |  |
| --- | --- |
| **Division** | **A** |
| **Batch** | **B2** |
| **GR-no** | **12110353** |
| **Roll no** | **10** |
| **Name** | **Aneesh Hemant Dighe** |

Line Encoding Techniques

CODE:

*DrawLinePanel1.java*

package computernetworks;

import java.awt.\*;

import javax.swing.JPanel;

public class DrawLinePanel1 extends JPanel{

private static final Color ***RED*** = null;

*@Override*

protected void paintComponent(Graphics g) {

// **TODO** Auto-generated method stub

super.paintComponent(g);

int c1=0;

int c2=50;

int c=0;

String s="1101100111";

int c3 = 25;

int c4 = 25;

g.setColor(***RED***);

//Bipolar RZ

g.drawString("Bipolar RZ Encoding Technique",750,100);

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 175, i, 35);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

String s1 = Character.*toString*(ch);

g.drawString(s1, c3, c4);

if(ch=='0')

{

g.drawLine(c1, 100,c2 , 100);

}

else

{

if(c%2==0)

{

g.drawLine(c1, 100, c1, 50);

g.drawLine(c1,50,c1+(c2-c1)/2,50);

g.drawLine(c1+(c2-c1)/2,50, c1+(c2-c1)/2,100);

g.drawLine(c1+(c2-c1)/2, 100, c2, 100);

}

else

{

g.drawLine(c1, 100, c1, 150);

g.drawLine(c1, 150, c1+(c2-c1)/2, 150);

g.drawLine(c1+(c2-c1)/2,150,c1+(c2-c1)/2,100);

g.drawLine(c1+(c2-c1)/2, 100, c2, 100);

}

c++;

}

c1+=50;

c2+=50;

c3+=50;

//c4+=25;

/\*g.drawLine(c1, 550, c1, 600);

g.drawLine(c1,600,c1+(c2-c1)/2,600);

g.drawLine(c1+(c2-c1)/2, 600, c1+(c2-c1)/2, 550);

g.drawLine(c1+(c2-c1)/2, 550,c2 ,550);\*/

//Manchester encoding

}

c1=0;

c2=50;

g.drawString("Manchester Encoding Thomas Technique",750,225);

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

String s1 = Character.*toString*(ch);

//g.drawString(s1, c3, c4);

if(ch=='1')

{

//g.drawLine(c1, 250,c1+(c2-c1)/2 , 200);

g.drawLine(c1, 200,c1+(c2-c1)/2 , 200);

g.drawLine(c1+(c2-c1)/2, 200, c1+(c2-c1)/2, 250);

g.drawLine(c1+(c2-c1)/2,250,c2,250);

if(i!=s.length()-1&&s.charAt(i)=='1'&&s.charAt(i+1)=='1')

{

g.drawLine(c2, 250, c2, 200);

}

}

else if(ch=='0')

{

g.drawLine(c1+(c2-c1)/2, 250, c1, 250);

g.drawLine(c1+(c2-c1)/2, 250, c1+(c2-c1)/2, 200);

g.drawLine(c1+(c2-c1)/2, 200, c2, 200);

if(i!=s.length()-1&&s.charAt(i)=='0'&&s.charAt(i+1)=='0')

{

g.drawLine(c2, 200, c2, 250);

}

}

c1+=50;

c2+=50;

c3+=50;

//c4+=25;

}

c1=0;

c2=50;

g.drawString("Manchester Encoding IEEE 802.3 Technique",750,325);

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

String s1 = Character.*toString*(ch);

//g.drawString(s1, c3, c4);

if(ch=='0')

{

//g.drawLine(c1, 250,c1+(c2-c1)/2 , 200);

g.drawLine(c1, 300,c1+(c2-c1)/2 , 300);

g.drawLine(c1+(c2-c1)/2, 300, c1+(c2-c1)/2, 350);

g.drawLine(c1+(c2-c1)/2,350,c2,350);

if(i!=s.length()-1&&s.charAt(i)=='0'&&s.charAt(i+1)=='0')

{

g.drawLine(c2,350, c2, 300);

}

}

else if(ch=='1')

{

g.drawLine(c1+(c2-c1)/2, 350, c1, 350);

g.drawLine(c1+(c2-c1)/2, 350, c1+(c2-c1)/2, 300);

g.drawLine(c1+(c2-c1)/2, 300, c2, 300);

if(i!=s.length()-1&&s.charAt(i)=='1'&&s.charAt(i+1)=='1')

{

g.drawLine(c2, 300, c2, 350);

}

}

c1+=50;

c2+=50;

}

//g.drawString(" Differential Manchester Encoding Thomas Technique",750,425);

// differencial manchester

g.drawString("Differential-Manchester Encoding: " , 800, 525);

int x1 = 0;

int x2 = 50;

int y = 500;

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 575, i, 400);

}

int last=0;

g.drawLine(x1, y, x1+500 , y);

for (int i = 0; i < s.length(); i++) {

char ch = s.charAt(i);

if(i==0){

if(ch=='0'){

g.drawLine(x1, y+50, x1, y - 50);

g.drawLine(x1, y - 50, x1 + 25, y - 50);

g.drawLine(x1 + 25, y - 50, x2 - 25, y+50);

g.drawLine(x1 + 25, y+50, x2, y+50);

last=-1;

}

else{

g.drawLine(x1, y-50, x2 - 25, y-50);

g.drawLine(x1 + 25, y-50, x1 + 25, y + 50);

g.drawLine(x1 + 25, y + 50, x2, y +50);

//g.drawLine(x2, y - 50, x2, y);

last=-1;

}

}else{

if (ch == '0') {

if(last==1){

g.drawLine(x1, y-50, x1, y + 50);

g.drawLine(x1, y + 50, x1 + 25, y + 50);

g.drawLine(x1 + 25, y + 50, x2 - 25, y-50);

g.drawLine(x1 + 25, y-50, x2, y-50);

last=1;

}else{

g.drawLine(x1, y+50, x1, y - 50);

g.drawLine(x1, y - 50, x1 + 25, y - 50);

g.drawLine(x1 + 25, y - 50, x2 - 25, y+50);

g.drawLine(x1 + 25, y+50, x2, y+50);

last=-1;

}

} else {

if(last==1){

g.drawLine(x1, y-50, x2 - 25, y-50);

g.drawLine(x1 + 25, y-50, x1 + 25, y + 50);

g.drawLine(x1 + 25, y + 50, x2, y +50);

last=-1;

}else{

g.drawLine(x1, y+50, x2 - 25, y+50);

g.drawLine(x1 + 25, y+50, x1 + 25, y - 50);

g.drawLine(x1 + 25, y - 50, x2, y -50);

last=1;

}

}

}

x1 += 50;

x2 += 50;

}

}

}

DrawLinePanel.java

package computernetworks;

import javax.swing.\*;

import java.awt.\*;

import java.util.\*;

public class DrawLinePanel extends JPanel{

@Override

protected void paintComponent(Graphics g)

{

super.paintComponent(g);

Scanner sc=new Scanner(System.in);

//System.out.println("Enter a string");

//String s=sc.next();

String s = "110110111";

//float[] dashPattern = { 2f, 2f }; // {Length of the dash, Length of the gap between dashes}

//BasicStroke dashedStroke = new BasicStroke(1f, BasicStroke.CAP\_BUTT, BasicStroke.JOIN\_MITER, 1f, dashPattern, 0f);

// Drawing Bipolar nRZ Encoding

int c1 = 0;

int c2 = 50;

int c3 = 25;

int c4 = 25;

int c = 0;

g.drawLine(0, 100, 500, 100); // Horizontal line for separation

g.drawString("Bipolar NRZ Encoding Technique", 750,100);

for (int i = 0; i < s.length(); i++) {

char ch = s.charAt(i);

String s1 = Character.toString(ch);

g.drawString(s1, c3, c4);

if (ch == '0') {

g.drawLine(c1, 100, c2, 100);

} else {

if (c % 2 == 0) {

g.drawLine(c1, 50, c1, 100);

g.drawLine(c1, 50, c2, 50);

g.drawLine(c2, 100, c2, 50);

} else {

g.drawLine(c1, 100, c1, 150);

g.drawLine(c1, 150, c2, 150);

g.drawLine(c2, 150, c2, 100);

}

c++;

}

c1 += 50;

c2 += 50;

c3 += 50;

}

// Drawing Unipolar NRZ Encoding

int c5 = 0;

c1 = 0;

c2 = 50;

// Drawing Unipolar Encoding

g.drawLine(0,250,500,250);

g.drawString("Unipolar Encoding Technique", 750,250);

for(int i=50;i<=500;i+=50) {

g.drawLine(i,150,i ,50 );

}

for(int i=50;i<=500;i+=50) {

g.drawLine(i,250,i ,200 );

}

c3=0;

for (int i = 0; i < s.length(); i++) {

char ch = s.charAt(i);

String s2 = Character.toString(ch);

if (ch == '0') {

g.drawLine(c1, 250, c2, 250); // Signal line for bit value '0' (low voltage)

} else {

g.drawLine(c1, 200, c1, 250); // Signal line for bit value '1' (high voltage)

g.drawLine(c1, 200, c2, 200);

if(i!=s.length()-1 && s.charAt(i)=='1'&&s.charAt(i+1)=='1')

{

c3=c2+50;

g.drawLine(c2, 200, c3, 200);

}

else

{

g.drawLine(c2, 250, c2, 200);

}

}

c1 += 50;

c2 += 50;

c3=0;// Move the starting position of the next bit's signal line

}

//Draw Unipolar NRZ Encoding

c1=0;

c2=50;

g.drawLine(0,350,500,350);

g.drawString("Unipolar NRZ Encoding Technique", 750,350);

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 350, i, 300);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

if(ch=='1')

{

g.drawLine(c1, 300, c1, 350);

g.drawLine(c1, 300, c2, 300);

g.drawLine(c2, 350, c2, 300);

c1+=50;

c2+=50;

}

else if(ch=='0')

{

c1+=50;

c2+=50;

}

}

//Draw Unipolar RZ Encoding

c1=0;

c2=50;

g.drawString("Unipolar RZ Encoding Technique", 750,450);

int c3333=50;

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 450, i, 400);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

if (ch == '0') {

g.drawLine(c1, 450, c2, 450); // Signal line for bit value '0' (low voltage)

} else {

g.drawLine(c1, 450, c1, 400); // Signal line for bit value '1' (high voltage)

g.drawLine(c1, 400, c1+(c2-c1)/2, 400);

g.drawLine(c1+(c2-c1)/2, 400, c1+(c2-c1)/2, 450);

g.drawLine(c1+(c2-c1)/2, 450, c2, 450);

}

System.out.println(c1+" "+c2);

c1+=50;

c2+=50;

}

//Polar RZ

c1=0;

c2=50;

g.drawString("Polar RZ Encoding Technique", 750,550);

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 600, i, 500);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

if(ch=='1')

{

g.drawLine(c1, 550, c1, 500);

g.drawLine(c1,500,c1+(c2-c1)/2,500);

g.drawLine(c1+(c2-c1)/2,500,c1+(c2-c1)/2,550);

g.drawLine(c1+(c2-c1)/2, 550, c2, 550);

}

else if(ch=='0')

{

g.drawLine(c1, 550, c1, 600);

g.drawLine(c1,600,c1+(c2-c1)/2,600);

g.drawLine(c1+(c2-c1)/2, 600, c1+(c2-c1)/2, 550);

g.drawLine(c1+(c2-c1)/2, 550,c2 ,550);

}

c1+=50;

c2+=50;

}

//Polar NRZ

c1=0;

c2=50;

g.drawString("Polar NRZ Encoding Technique", 750,700);

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 775, i, 625);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

if(ch=='1')

{

g.drawLine(c1, 650, c1, 750);

g.drawLine(c1, 650, c2, 650);

g.drawLine(c2, 750, c2, 650);

}

else

{

g.drawLine(c1, 750, c2, 750);

}

c1+=50;

c2+=50;

}

DrawLine.java

package computernetworks;

import javax.swing.\*;

public class DrawLine {

public static void main(String args[])

{

JFrame fr=new JFrame("Draw a line");

fr.setSize(1500,1500);

DrawLinePanel panel=new DrawLinePanel();

DrawLinePanel1 panel1=new DrawLinePanel1();

//fr.add(panel);

fr.add(panel1);

fr.setVisible(true);

}

}

Bipolar RZ

g.drawString("Bipolar RZ Encoding Technique",750,100);

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 175, i, 35);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

String s1 = Character.*toString*(ch);

g.drawString(s1, c3, c4);

if(ch=='0')

{

g.drawLine(c1, 100,c2 , 100);

}

else

{

if(c%2==0)

{

g.drawLine(c1, 100, c1, 50);

g.drawLine(c1,50,c1+(c2-c1)/2,50);

g.drawLine(c1+(c2-c1)/2,50, c1+(c2-c1)/2,100);

g.drawLine(c1+(c2-c1)/2, 100, c2, 100);

}

else

{

g.drawLine(c1, 100, c1, 150);

g.drawLine(c1, 150, c1+(c2-c1)/2, 150);

g.drawLine(c1+(c2-c1)/2,150,c1+(c2-c1)/2,100);

g.drawLine(c1+(c2-c1)/2, 100, c2, 100);

}

c++;

}

c1+=50;

c2+=50;

c3+=50;

//c4+=25;

/\*g.drawLine(c1, 550, c1, 600);

g.drawLine(c1,600,c1+(c2-c1)/2,600);

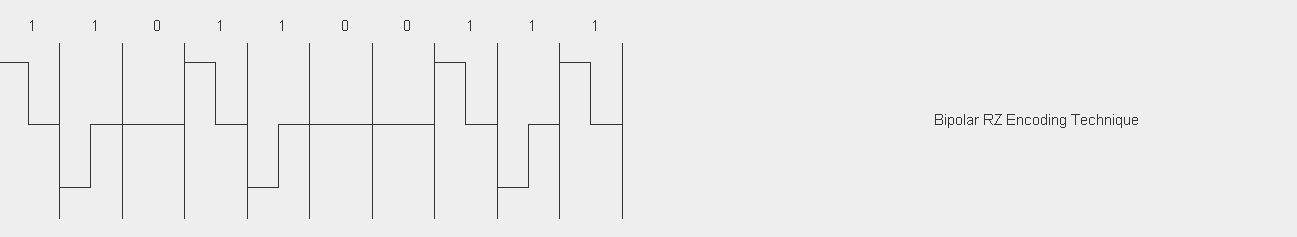
g.drawLine(c1+(c2-c1)/2, 600, c1+(c2-c1)/2, 550);

g.drawLine(c1+(c2-c1)/2, 550,c2 ,550);\*/

//Manchester encoding

}

OUTPUT:



Bipolar NRZ

g.drawLine(0, 100, 500, 100); // Horizontal line for separation

g.drawString("Bipolar NRZ Encoding Technique", 750,100);

for (int i = 0; i < s.length(); i++) {

char ch = s.charAt(i);

String s1 = Character.*toString*(ch);

g.drawString(s1, c3, c4);

if (ch == '0') {

g.drawLine(c1, 100, c2, 100);

} else {

if (c % 2 == 0) {

g.drawLine(c1, 50, c1, 100);

g.drawLine(c1, 50, c2, 50);

g.drawLine(c2, 100, c2, 50);

} else {

g.drawLine(c1, 100, c1, 150);

g.drawLine(c1, 150, c2, 150);

g.drawLine(c2, 150, c2, 100);

}

c++;

}

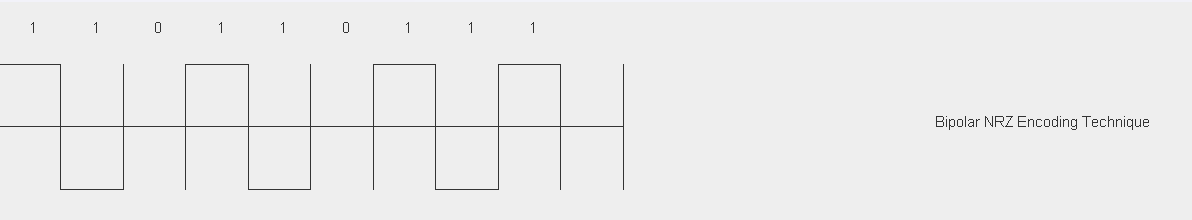
c1 += 50;

c2 += 50;

c3 += 50;

}

OUTPUT:



POLAR NRZ:

c1=0;

c2=50;

g.drawString("Polar NRZ Encoding Technique", 750,700);

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 775, i, 625);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

if(ch=='1')

{

g.drawLine(c1, 650, c1, 750);

g.drawLine(c1, 650, c2, 650);

g.drawLine(c2, 750, c2, 650);

}

else

{

g.drawLine(c1, 750, c2, 750);

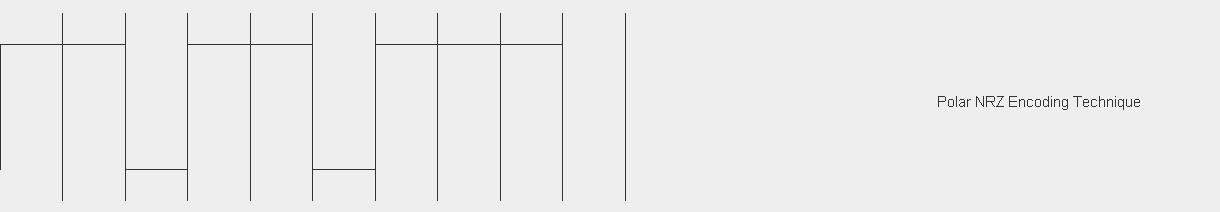
}

c1+=50;

c2+=50;

}

OUTPUT:



POLAR RZ:

c1=0;

c2=50;

g.drawString("Polar RZ Encoding Technique", 750,550);

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 600, i, 500);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

if(ch=='1')

{

g.drawLine(c1, 550, c1, 500);

g.drawLine(c1,500,c1+(c2-c1)/2,500);

g.drawLine(c1+(c2-c1)/2,500,c1+(c2-c1)/2,550);

g.drawLine(c1+(c2-c1)/2, 550, c2, 550);

}

else if(ch=='0')

{

g.drawLine(c1, 550, c1, 600);

g.drawLine(c1,600,c1+(c2-c1)/2,600);

g.drawLine(c1+(c2-c1)/2, 600, c1+(c2-c1)/2, 550);

g.drawLine(c1+(c2-c1)/2, 550,c2 ,550);

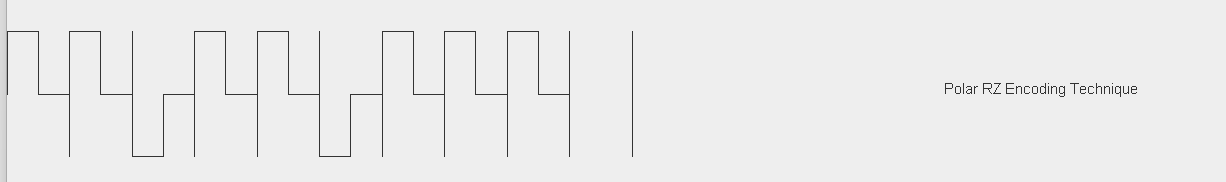
}

c1+=50;

c2+=50;

}

OUTPUT:



UNIPOLAR NRZ:

c1=0;

c2=50;

g.drawLine(0,350,500,350);

g.drawString("Unipolar NRZ Encoding Technique", 750,350);

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 350, i, 300);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

if(ch=='1')

{

g.drawLine(c1, 300, c1, 350);

g.drawLine(c1, 300, c2, 300);

g.drawLine(c2, 350, c2, 300);

c1+=50;

c2+=50;

}

else if(ch=='0')

{

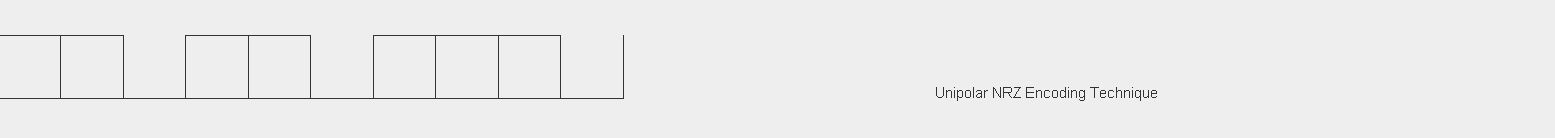
c1+=50;

c2+=50;

}

}

OUTPUT:



UNIPOLAR RZ:

c1=0;

c2=50;

g.drawString("Unipolar RZ Encoding Technique", 750,450);

int c3333=50;

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 450, i, 400);

}

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

if (ch == '0') {

g.drawLine(c1, 450, c2, 450); // Signal line for bit value '0' (low voltage)

} else {

g.drawLine(c1, 450, c1, 400); // Signal line for bit value '1' (high voltage)

g.drawLine(c1, 400, c1+(c2-c1)/2, 400);

g.drawLine(c1+(c2-c1)/2, 400, c1+(c2-c1)/2, 450);

g.drawLine(c1+(c2-c1)/2, 450, c2, 450);

}

System.***out***.println(c1+" "+c2);

c1+=50;

c2+=50;

}

OUTPUT:



MANCHESTER IEEE 802.3 TECHNIQUE:

c1=0;

c2=50;

g.drawString("Manchester Encoding IEEE 802.3 Technique",750,325);

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

String s1 = Character.*toString*(ch);

//g.drawString(s1, c3, c4);

if(ch=='0')

{

//g.drawLine(c1, 250,c1+(c2-c1)/2 , 200);

g.drawLine(c1, 300,c1+(c2-c1)/2 , 300);

g.drawLine(c1+(c2-c1)/2, 300, c1+(c2-c1)/2, 350);

g.drawLine(c1+(c2-c1)/2,350,c2,350);

if(i!=s.length()-1&&s.charAt(i)=='0'&&s.charAt(i+1)=='0')

{

g.drawLine(c2,350, c2, 300);

}

}

else if(ch=='1')

{

g.drawLine(c1+(c2-c1)/2, 350, c1, 350);

g.drawLine(c1+(c2-c1)/2, 350, c1+(c2-c1)/2, 300);

g.drawLine(c1+(c2-c1)/2, 300, c2, 300);

if(i!=s.length()-1&&s.charAt(i)=='1'&&s.charAt(i+1)=='1')

{

g.drawLine(c2, 300, c2, 350);

}

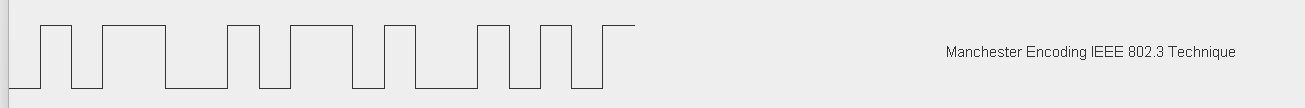
}

c1+=50;

c2+=50;

}

OUTPUT:



MANCHESTER THOMAS TECHNIQUE:

c1=0;

c2=50;

g.drawString("Manchester Encoding Thomas Technique",750,225);

for(int i=0;i<s.length();i++)

{

char ch=s.charAt(i);

String s1 = Character.*toString*(ch);

//g.drawString(s1, c3, c4);

if(ch=='1')

{

//g.drawLine(c1, 250,c1+(c2-c1)/2 , 200);

g.drawLine(c1, 200,c1+(c2-c1)/2 , 200);

g.drawLine(c1+(c2-c1)/2, 200, c1+(c2-c1)/2, 250);

g.drawLine(c1+(c2-c1)/2,250,c2,250);

if(i!=s.length()-1&&s.charAt(i)=='1'&&s.charAt(i+1)=='1')

{

g.drawLine(c2, 250, c2, 200);

}

}

else if(ch=='0')

{

g.drawLine(c1+(c2-c1)/2, 250, c1, 250);

g.drawLine(c1+(c2-c1)/2, 250, c1+(c2-c1)/2, 200);

g.drawLine(c1+(c2-c1)/2, 200, c2, 200);

if(i!=s.length()-1&&s.charAt(i)=='0'&&s.charAt(i+1)=='0')

{

g.drawLine(c2, 200, c2, 250);

}

}

c1+=50;

c2+=50;

c3+=50;

//c4+=25;

}

OUTPUT:



DIFFERENTIAL MANCHESTER ENCODING TECHNIQUE:

g.drawString("Differential-Manchester Encoding: " , 800, 525);

int x1 = 0;

int x2 = 50;

int y = 500;

for(int i=50;i<=500;i+=50)

{

g.drawLine(i, 575, i, 400);

}

int last=0;

g.drawLine(x1, y, x1+500 , y);

for (int i = 0; i < s.length(); i++) {

char ch = s.charAt(i);

if(i==0){

if(ch=='0'){

g.drawLine(x1, y+50, x1, y - 50);

g.drawLine(x1, y - 50, x1 + 25, y - 50);

g.drawLine(x1 + 25, y - 50, x2 - 25, y+50);

g.drawLine(x1 + 25, y+50, x2, y+50);

last=-1;

}

else{

g.drawLine(x1, y-50, x2 - 25, y-50);

g.drawLine(x1 + 25, y-50, x1 + 25, y + 50);

g.drawLine(x1 + 25, y + 50, x2, y +50);

//g.drawLine(x2, y - 50, x2, y);

last=-1;

}

}else{

if (ch == '0') {

if(last==1){

g.drawLine(x1, y-50, x1, y + 50);

g.drawLine(x1, y + 50, x1 + 25, y + 50);

g.drawLine(x1 + 25, y + 50, x2 - 25, y-50);

g.drawLine(x1 + 25, y-50, x2, y-50);

last=1;

}else{

g.drawLine(x1, y+50, x1, y - 50);

g.drawLine(x1, y - 50, x1 + 25, y - 50);

g.drawLine(x1 + 25, y - 50, x2 - 25, y+50);

g.drawLine(x1 + 25, y+50, x2, y+50);

last=-1;

}

} else {

if(last==1){

g.drawLine(x1, y-50, x2 - 25, y-50);

g.drawLine(x1 + 25, y-50, x1 + 25, y + 50);

g.drawLine(x1 + 25, y + 50, x2, y +50);

last=-1;

}else{

g.drawLine(x1, y+50, x2 - 25, y+50);

g.drawLine(x1 + 25, y+50, x1 + 25, y - 50);

g.drawLine(x1 + 25, y - 50, x2, y -50);

last=1;

}

}

}

x1 += 50;

x2 += 50;

}

OUTPUT:

